<u>REMARKS</u>

Claims 1-68 remain pending in the application. Reconsideration is respectfully requested in light of the following remarks.

<u>Section 102(e) Rejection</u>:

The Office Action rejected claims 1-5, 11-17, 22-28, 34-37, 42-46, 48-52, 54, 49, 63-65 and 68 under 35 U.S.C. § 102(e) as being anticipated by Bittenger et al. (U.S. Patent 6,453,362) (hereinafter "Bittenger"). Applicants respectfully traverse this rejection for at least the following reasons.

Regarding claim 1, the Examiner states that Bittenger discloses "remotely invoking methods in a distributed computing environment, comprising: A client generating a message in a data representation language, wherein the message includes information representing a computer programming language method call, and wherein the message further includes a credential for allowing the client access to a service configured to perform functions on behalf of clients in the distributed computing environment … if said examining determines the credential is authentic, the service performing a function on behalf of the client in accordance with the information representing the computer programming language method call included in the message." Applicants respectfully disagree with the Examiner's interpretation of Bittenger.

Bittenger teaches a method wherein a client invokes a remote server application using a server stub object that packs or marshals method parameters into a request (Bittenger, column 6, lines 9-17, and lines 28-33). Bittenger also discloses the use of systems such as CORBA, DCOM, RMI and Java to invoke methods on remote server applications. Note that CORBA, DCOM, RMI and Java are all code-centric technologies *that by definition do not employ data representation language messages for method calls.* Bittenger does not teach a client generating <u>a message in a data representation language</u>, wherein the message includes information <u>representing</u> a computer programming language method call. Instead, Bittenger teaches remote object communications using code-centric technologies such as CORBA, DCOM and RMI that employ programmatic method invocations.

In response to this argument, the Examiner states on p. 23 of the Final Action that "a data representation language seems to be nothing more than a method of encoding data for transmission between a client and a server where both the transmitter and receiver have means of reading the data encoded in the message." The Examiner's interpretation of generating a message in a data representation language representing a computer programming language method call is clearly incorrect. Generating a message in a data representation language representing a computer programming language method call is not merely a method of encoding data for transmission between a client and a server, and is distinctly different from remote method invocations using, for example, stubs, RMI or another code-based communication format. Note that the code-centric nature of prior art remote interfaces such as those used in Bittenger (e.g. CORBA and RMI) is discussed in Applicants' Related Art section on pp. 8-10 of the present application. In contrast, data representation languages are known in the art as languages that are used to represent or describe data in documents. The eXtensible Markup Language is one example of a data representation language. Data representation languages are not used in the prior art for programmatic interfaces such as the stub in Bittenger. Thus, Bittenger clearly does not anticipate generating a message in a data representation language representing a computer programming language method call.

In contrast, Bittenger describes a remote communication mechanism employing stubs and based on technologies such as CORBA or RMI. In the prior art, such mechanisms are specifically <u>not</u> based on data representation languages. A data representation language is a specific type of language having a specific purpose for describing data documents, as is well known to those of ordinary skill in the art. In the prior art, data representation languages were only used to describe data documents or content (such as HTML or XML documents that may be displayed or processed in Internet communications). In Bittenger, clients use a programmatic (code-based) remote interface to invoke services. The communications in Bittenger are clearly not data representation language messages. Furthermore, in Bittenger stubs are used to make an actual remote call. In other words, the client stub communications are actual CORBA or RMI code-centric method calls, not messages <u>representing</u> method calls.

Additionally, Bittenger teaches the use of a ticket object created by the client that a server application can use to pass a server stub object to the client. The client can then use the server stub object to invoke remote methods on the server application (Bittenger, column 6, lines 63-67, column 7, lines 42-47, and column 8, lines 15-22). Hence, under Bittenger, the ticket is used to notify the client that the server application is running and ready to receive requests and provides a method for the server application to provide a server stub object to the client. Further, Bittenger teaches the user of a separate authentication server to authenticate the client using well-known login procedures such as user id and password. After authenticating the client, the authentication server then forwards the ticket to the server application (Bittenger, column 7, line 67 – column 8, line 8). Therefore Applicants assert that Bittenger does not teach the use of a ticket as a credential as suggested by the Examiner. The ticket in Bittenger is not in a message with a representation of a method call for a function of a service. The login procedure referred to by the Examiner in his Response to Arguments section has nothing to do with including a credential with a representation of a method call for a function of a service.

Thus, Applicants submit that Bittenger fails to disclose a method for remotely invoking methods in a distributed computing environment, comprising: A client <u>generating a message in a data representation language</u>, wherein the message includes information <u>representing a computer programming language method call</u>, and wherein the <u>message further includes a credential</u> for allowing the client access to a service configured to perform functions on behalf of clients in the distributed computing environment.

Further regarding claim 1, the Examiner states, "Bittenger discloses a method for remotely invoking methods in a distributed computing environment, comprising … [t]he service examining the credential included in the message." Applicants respectfully disagree with the Examiner's interpretation of Bittenger.

In contrast, Bittenger teaches the use of a separate server authenticating a client prior to the launching of the desired server application (Bittenger, column 8, lines 9-14). Bittenger fails to teach that any credential is verified by the service application that performs the function in accordance with the representation of the method call. Thus, applicants submit that Bittenger does not disclose a method comprising a <u>service examining a credential included in the message</u>.

In light of the above remarks, Applicants assert that the rejection of claim 1 is not supported by the cited art and withdrawal of the rejection is respectfully requested. The rejection of independent claims 25, 45, 49 and 59 is unsupported by the cited art for similar reasons as discussed above.

Regarding claim 5, the Examiner states that Bittenger discloses "the client message endpoint attaching the credential to the message (tStamp is an identifier used on all messages)." Applicant disagrees with the Examiner.

Bittenger teaches that "the identifier 'tstamp' may be used to locate the ticket within the [client-side registry] database" (Bittenger, column 7, lines 8-9). Bittenger further describes a server using the tstamp to retrieve a ticket stub from the client-side registry (Bittenger, column 7, lines 41-43). Contrary to the Examiner's assertion, Bittenger does not teach that a tstamp is used on all messages. Further, Bittenger's use of a tstamp to allow a server application to locate and retrieve a ticket stub from a client-side registry does not constitute a client message endpoint attaching the credential to the message.

In light of the above remarks, Applicants assert that the rejection of claim 5 is not supported by the cited art and withdrawal of the rejection is respectfully requested. Similar arguments apply in regard to claim 28.

In regard to claim 13, Bittenger does not teach information representing the computer programming language method call (in a data representation language message) including an identifier of the method call, and wherein said performing a function comprises regenerating the method call in accordance with the identifier of the method call included in the information representing the method call, and executing a computer programming language method of the service in accordance with the regenerated method call. In Bittenger, a client stud is used to make a remote call to a service using CORBA, RMI or a similar technology. As is well understood by anyone of ordinary skill in the art, in technologies such CORBA and RMI a method call is made remotely. CORBA and RMI doe not involve sending any representation of the method call that is then regenerated into the method call. Thus, claim 13 is clearly not anticipated by Bittenger. Similar arguments apply in regard to claims 34 and 63.

Section 103(a) Rejection:

The Office Action rejected claims 18-20, 38-40, 47, 53 and 66 under 35 U.S.C. § 103(a) as being unpatentable over Bittenger in view of Leach, et al. (U.S. Patent 6,108,715) (hereinafter "Leach"), claims 21, 41 and 67 were unpatentable over Bittenger in view of the Instaweb Online Computing Dictionary (Instaweb, htttp://www.instantweb.com/foldoc/foldoc.cgi?query=XML) (hereinafter "Instaweb").

In regard to claim 18, the cited art does not teach or suggest storing the generated results data to a space service in the distributed computing environment, providing an advertisement for the stored results data to the client, wherein the advertisement comprises information to enable access by the client to the stored results data, and the client accessing the stored results data from the space service in accordance with the information in the provided advertisement. The Examiner states, "Leach discloses the creation of a data stack used to store the results of the operations locally on the server, then allows the client to map back to the stored results stack by providing an address which allows direct transfer between the client and the server greatly reducing the processing overhead." Applicants fail to see the relevance of Leach's teachings to the limitations of claim 18. Furthermore, Applicants respectfully disagree with the Examiner's interpretation of Leach.

Leach teaches direct stack-to-stack transfer of parameters between two processes executing concurrently on a single machine. Rather than marshaling and unmarshaling parameters when invoking a remote procedure call in a second process, Leach teaches the use of a shared data stack and that a operating system kernel process copies the input and output parameters between the two processes individual stacks. (Leach, column 4, lines 32-43). Further, Leach teaches directly copying data from a server process to a client process through the use of a kernel that can access the address spaces of each process, thereby facility such a direct transfer to data (Leach, column 5, lines 44-52). Thus, contrary to the Examiner's assertion, Leach does not disclose the creation of a data stack used to store the results of the operations locally on the server and then allows the client to map back to the stored results stack by providing an address.

Specifically, under Leach, the client is not provided an address to map back to the any stored results stack, as the Examiner contends. Instead, Leach teaches that a kernel process directly transfers the output parameters onto the client's stack and "transfers control to the client process with the instruction pointer point to the instruction that follows the client process' call to the proxy method" (Leach, column 5, lines 52-55). Hence, Leach's kernel transfers any results directly into the client's stack and does not provide the client any address or information that would allow the client to "map back" to the stored results stack. Further, Leach teaches that the kernel must do this, because the client would not be able to access the other process' stack (Leach, column 6, lines 51-54).

Bittenger in view of Leach clearly does no teach or suggest storing the generated results data to a space service in the distributed computing environment, providing an advertisement for the stored results data to the client, wherein the advertisement comprises information to enable access by the client to the stored results data, and the client accessing the stored results data from the space service in accordance with the information in the provided advertisement. The rejection of claim 18 is not supported by the cited art and withdrawal of the rejection is respectfully requested. Similar arguments apply in regard to claims 38, 47, 53 and 66.

Applicants also assert that numerous ones of the other dependent claims recite further distinctions over the cited art. However, since the independent claims have been shown to be patentably distinct, a further discussion of the dependent claims is not necessary at this time.

Allowable Subject Matter:

Claims 6-10, 19-33 and 60-62 were objected to as being dependent upon a rejected base claim, but otherwise allowable if rewritten in independent form. In light of the above remarks, Applicants assert that claims 6-10, 19-33 and 60-62 are allowable as depending from patentably distinct base claims. Applicants therefore respectfully request allowance of claims 6-10, 19-33 and 60-62 as currently pending.

Allowed Claims:

Claims 55-58 have been allowed.

Information Disclosure Statement:

The Examiner states that three information disclosure statements have been reviewed. However, four information disclosure statements have been submitted. Specifically, Applicants have not received back from the Examiner the signed and initialed copies of the Forms PTO-1449 from the information disclosure statement submitted on August 17, 2001. A copy of the previously submitted Forms PTO-1449 from the information disclosure statement submitted on August 17, 2001 are included herewith for the Examiner's convenience.

O I P E JC55
AUG 1 6 2004
PATENT & TRADEMARK OFFICE

CONCLUSION

Applicants submit the application is in condition for allowance, and notice to that effect is respectfully requested.

If any fees are due, the Commissioner is authorized to charge said fees to Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C. Deposit Account No. 501505/5181-67300/RCK.

Also enclosed herewith are the following items:

☒ Return Receipt Postcard

☐ Petition for Extension of Time

☐ Notice of Change of Address

☐ Fee Authorization Form authorizing a deposit account debit in the amount of $ for fees ( ).

☒ Copy of the previously submitted Forms PTO-1449 from the information disclosure statement submitted on August 17, 2001.

Respectfully submitted,

[illegible]

Robert C. Kowert
Reg. No. 39,255
ATTORNEY FOR APPLICANT(S)

Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C.
P.O. Box 398
Austin, TX 78767-0398
Phone: (512) 853-8850

Date: August 13, 2004

RECEIVED
AUG 1 8 2004
Technology Center 2100